CS 134: Networks — Problem set 10

Prof. Yaron Singer — Due: **Due Wednesday, 4/19/17 at noon**

The first two questions of this problem set are drawn from *Understanding Machine Learning: Theory and Algorithms.*

1. Suboptimality of k-means (Understanding Machine Learning, 22.8 Q1) (20 points) For every parameter $t > 1$, show that there exists an instance of the k-means problem for which the k-means algorithm (might) find a solution whose k-means objective is at least $t \cdot OPT$, where OPT is the minimum k-means objective.

2. k-means Might Not Necessarily Converge to a Local Minimum (Understanding Machine Learning, 22.8 Q2) (20 points) Show that the k-means algorithm might converge to a point which is not a local minimum. Hint: Suppose that $k = 2$ and the sample points are $\{1, 2, 3, 4\} \subset \mathbb{R}$; suppose we initialize the k-means with the centers $\{2, 4\}$; and suppose we break ties in the definition of C_i by assigning i to be the smallest value in $\text{argmin}_j \mid\mid x - \mu_j \mid\mid$.

3. Ethics, revisited (30 points) During the ethics lecture, we discussed a number of different strategies that Facebook could use to suppress the spread of fake news through its network. Briefly explain one of these strategies in a single paragraph. Then, in two further paragraphs, explain whether or not you think Facebook is morally obligated to implement the strategy. Defend your answer, drawing on material from the ethics lecture. Your answer should be between 400 and 600 words.

4. Investigating Parameters of K-Means (30 points) In this problem you will investigate the use of various parameters of k-means, in particular the number of clusters k and methods of assigning nodes to clusters.

a. **(5 points)** Load the undirected graph from the file `data.txt`. How many nodes and edges does this network have?

b. **(6 points)** Implement a function `min_pairwise(g)` that takes a graph and returns a new matrix `distances` of size `(n,n)` where `n` is the number of nodes in the graph. This matrix `distances` should describe the shortest path distances between any two nodes, for example `distances[i][j]` should be the shortest distance from `i` to `j` on the graph. What's the average shortest pairs distance over all pairs?

   Hint: You can do this several ways. One option uses BFS and should have complexity $O(V(V + E))$ (that is, it'll iterate over [every node and edge] for every node); while another BFS option will have complexity $O(E(V + E))$; and another will use an algorithm called Floyd-Warshall and have complexity $O(V^3)$; only one of these options will be moderately fast, and thus, correct! **However you do it, do it yourself without using any modules or packages.**

c. **(3 points)** Implement a function `dist(p, m, c, norm)` that accepts a node `p`, a matrix of shortest distances `m`, a "cluster" (an array or set, your choice) of nodes `c`, and a string parameter `norm` with possible values "min", "max", or "mean" referring to the following respective metrics. This function should return the appropriate distance between the point and the cluster `c`:

norm	metric
`min(p,c)`	$\min_{i \in c}\{m[p][c]\}$
`max(p,c)`	$\max_{i \in c}\{m[p][c]\}$
`mean(p,c)`	$\frac{\sum_{i \in c}\{m[p][c]\}}{\lvert c \rvert}$

What's the distance between node 5 and the cluster $\{2, 8, 20\}$ under each of the three metrics?

d. **(2 points)** Write a function `assign(p, m, c_list, norm)` that accepts a node `p`, a matrix of shortest distances `m` and a list of clusters `c_list`, and a string parameter `norm` as defined above. This function should utilize `dist` to find the closest cluster to `p` and return the index of that cluster in the list of clusters `c_list`.

Given node 5 and clusters `[{2, 8, 20}, {3, 4, 8, 26}]`, what does `assign` return for each of the three metrics?

e. **(3 points)** Write a function `center(m, c)` that accepts a matrix of shortest distances `m` and a cluster `c`, and returns the node that minimizes the k-means objective function within the cluster `c`, i.e.:

$$\underset{i \in c}{\operatorname{argmin}} \sum_{j \in c} (m[j][i])^2$$

Given cluster `{2, 3, 4, 8, 20, 26}`, what node is the center of the cluster?

f. **(6 points)** Time to put it all together! Write a function `cluster(m, k, norm, i)` that accepts a shortest distances matrix `m`, a number of clusters `k`, a string parameter `norm` as defined above, and a number of iterations `i`. This function should be your k-means implementation, and should return a list of the clusters you obtain. Your k-means algorithm should follows these steps, which **differs in specific ways from the canonical version**:

(a) Randomly select k nodes to initialize the clusters.

(b) In random order, assign every other node in the graph to one of the clusters.

(c) Reinitialize new clusters at the "center" nodes of each of the clusters.

(d) Repeat steps (b) and (c) as many times as required/desired.

Note that the standard k-means algorithm works on points in space, and computes assignments using means of the clusters. We here instead apply k-means to networks using shortest edge distance and assign based on entire clusters instead of precalculated single node centers. Both of these modifications remove the guarantee that this algorithm will converge, though the latter helps us minimize the objective function within fewer iterations (at the cost of increased computation time). As a result, you should cap the number of iterations you do at 20.

g. **(5 points)** Report the value of `center`, the value of objective function on the center, and the size of each cluster for 3 runs (max. 20 iterations each) of the k-means algorithm for each of the values of $k = [3, 5, 10, 20]$ for each of the three types of norms. (A table would be a good way to organize this data, and thus earn your TF's love.) What do you notice about the center/objective function and the sizes of your clusters as k increases and across different norms? Any guesses as to why this is? **Include your values for this part and all previous parts in your writeup for credit. Given the restrictions on iteration size and the trials we ask you to run, runtime should not be more than 40 minutes. Of particular computational cost are parts b. and g.**